



# UNIVERSITY OF BIRMINGHAM

## Named Entity Recognition in the WSJ

T.A. Goodman - **1526322**

*A brief discussion on the task of Named Entity Recognition in the Wall Street Journal (WSJ) and analysis of the resulting solution to the problem, created using Python 3 and the nltk toolkit.*

### 1 Outlining the Task

The given task was to use 2000 tagged Wall Street Journal (WSJ) articles to train a tagger that would then be able to tag 2000 untagged WSJ articles accurately in the same way that the tagged journals were tagged. The corpus was tagged specifically with three categories of Named Entities (NEs): Organizations, Locations and People. Each tag was of the form

`< ENAMEX TYPE = "type goes here" > Named Entity < /ENAMEX >`

and similarly, tags in this form would have to be added to the untagged data.

### 2 Preprocessing

Firstly, it was imperative to ensure that the training data was in a manageable format - A python script, `concat.py`, that concatenated the text from all 2000 files was used to siphon the data into a single file, `concatstr.txt`. Later on, this was modified to train the tagger on both a half, and a quarter of the given data.

This data is then split into words using regex, and each occurrence of a tag is placed into one of three Counter objects, for organizations, locations and people respectively. As well as the standard data, a second corpus of text that was similarly tagged was used in order to increase the accuracy of the tagger<sup>[1]</sup>.

### 3 Greedy N-Gram Tagger

The initial approach to the problem was a tagger that would go through the entire untagged corpus finding phrases of length  $n$ , attempting to tag these based on statistical occurrence in the three counters. Upon reaching the end of the corpus (i.e. no more  $n$ -length phrases could be formed), the tagger would back-off to a tagger that would tag for  $(n-1)$ -length phrases and so on until  $n = 0$ . This was basically a from-scratch implemented  $n$ -gram<sup>[2]</sup> tagger, and had a seemingly alright accuracy (though nothing precise was recorded).

## 4 Part-of-Speech Tagging & Chunking

In order to deal with unseen phrases, the next step was to implement shallow parsing (or chunking) in order to group Proper Nouns (NNPs) into manageable, likely-tagable phrases. The first step was to Part-of-Speech (POS)-tag[3] the untagged data. This was done using the `nlk.pos.tag()` function.

Once the data was POS-tagged, the next step was to chunk the NNP phrases. In order to do this, a function called *takewhileNNP()* was created, that would, in  $O(n)$  time, return the longest group of NNPs it could from the given data and a starting point. This group was then concatenated and encapsulated with tags:

$\langle ENAMEX\ TYPE = "untagged" \rangle NE \langle /ENAMEX \rangle$

In order to save time, the only phrases considered for tagging by the tagger were ENAMEX 'untagged' phrases.

## 5 Powerset Considerations

In the case of unseen phrases, the score assigned by the tagger would be 0, meaning that the phrase would remain untagged as it could not be categorised. In order to circumvent this, if a phrase scored 0 on each category, a 'powerset'[4] would be created from it - i.e.

$powerset(["This", "is", "NLP"]) \Rightarrow$

$["", "This", "is", "NLP", "This is", "is NLP", "This is NLP"]$

By scoring each of these sub-phrases and taking the highest score as it's category, it was then possible to assign categories to previously unseen words and phrases.

## 6 Evaluation of the tagger

In order to evaluate the tagger, 3 tests were conducted. In each case, the tagger was trained on different proportions of the training data: 100%, 50% and 25% respectively. The results were as follows:

% Trained	Accuracy
100	58.99%
50	51.24%
25	43.44%

It is worth noting that the actual accuracy will be slightly, if not notably higher than the evaluation code suggests, due to some issues with punctuation handling in both parts of the code.

## 7 Future Considerations & Conclusion

The next step in tagging the data would be wikification (or Entity Linking)[5] using an API such as DBPedia or Wikify in order to better-categorise the unknown phrases.

In conclusion, the final system is able to produce tagged-data that is of a relatively high standard, somewhat comparable to the gold-standard tagged data. With more work, it would be possible to iron-out a large swathe of the current issues with the system and implement further NLP-techniques in order to improve the accuracy even more.

## References

- [1] <https://github.com/dlwh/epic/blob/master/src/main/resources/ner/en.txt>
- [2] <https://en.wikipedia.org/wiki/N-gram>
- [3] [https://en.wikipedia.org/wiki/Part-of-speech\\_tagging](https://en.wikipedia.org/wiki/Part-of-speech_tagging)
- [4] [https://en.wikipedia.org/wiki/Power\\_set](https://en.wikipedia.org/wiki/Power_set)
- [5] [https://en.wikipedia.org/wiki/Entity\\_linking](https://en.wikipedia.org/wiki/Entity_linking)